If the reference picture is of a field interpolation mode, the index [] of the matrices mentioned above takes only select_mv_forward. If the reference picture is of a frame interpolation mode, the index [] of the matrices mentioned above takes both select_mv_forward and !select_mv_forward.

A positive value for a reconstructed horizontal motion vector (right_for[]) indicates that the referenced area of the past picture is to the right of the macroblock in the coded picture.

A positive value for a reconstructed vertical motion vector (down_for[]) indicates that the referenced area of the past picture is below the macroblock in the coded picture.

Defining pel_past[][][] as the pels of the past picture referenced by the forward motion vector, and pel[][][] as the pels of one field of the picture being decoded, then:

```
        if ( ! right_half_for[] && ! down_half_for[] )
                pel[][i][j] = pel_past[][i+down_for][j+right_for] ;

        if ( ! right_half_for[] &&  down_half_for[] )
                pel[][i][j] = ( pel_past[][i+down_for][j+right_for] +
                                        pel_past[][i+down_for+1][j+right_for] ) // 2
;

        if (  right_half_for[] && ! down_half_for[] )
                pel[][i][j] = ( pel_past[][i+down_for][j+right_for] +
                                        pel_past[][i+down_for][j+right_for+1] ) // 2
;

        if (  right_half_for[] &&  down_half_for[] )
                pel[][i][j] = ( pel_past[][i+down_for][j+right_for] +
                                pel_past[][i+down_for+1][j+right_for] +
                                pel_past[][i+down_for][j+right_for+1] +
                                pel_past[][i+down_for+1][j+right_for+1] ) // 4 ;
```

If the reference picture is of a field interpolation mode, the first index of pel_past[][][] and pel[][][] take only select_mv_forward. If the reference picture is frame interpolation mode, the first index of pel_past[][][] and pel[][][] takes both select_mv_forward and !select_mv_forward.

Define pel_new[][] as the pels of the picture being decoded, then:

```
        if ( field_or_frame_forward == 0 )
                pel_new[i][j] = pel[select_mv_forward][i][j] ;
        else
                pel_new[i][j] = ( pel[select_mv_forward][i][j] +
                                        pel[!select_mv_forward][i][j] ) // 2 ;
```

The DCT coefficients for each block present in the macroblock are reconstructed by:

```
        for ( m=0; m<8; m++ ) {
```

```
        for ( n=0; n<8; n++ ) {
            i = scan[][m][n] ;
            dct_recon[m][n] = ( 2 * dct_zz[i] * ( 16 + quantizer_scale
                                            * non_intra_quant[m][n] ) ) / 16 ;
            if ( ( dct_recon[m][n] & 1 ) == 0 )
                dct_recon[m][n] = dct_recon[m][n] - Sign(dct_recon[m][n])
;
            if (dct_recon[m][n] > 2047)  dct_recon[m][n] = 2047 ;
            if (dct_recon[m][n] < -2048)  dct_recon[m][n] = -2048 ;
            if ( dct_zz[i] == 0 )
                dct_recon[m][n] = 0 ;
        }
    }
```

If the reconstructed block is a luminance block, then the first index of scan[][][] shall be 2. If the reconstructed block is a chrominance block, then the first index of scan[][][] shall be 3.

NOTE - dct_recon[m][n] = 0 (for all m, n) in skipped macroblocks and when pattern[i] == 0.

Once the dct coefficients are reconstructed, the inverse DCT transform is applied to obtain the inverse transformed pel values in the interval [-256, 255]. The inverse DCT pel values are added to the pel_new[i][j] which were computed above using the motion vectors, with the result of the addition being limited to the interval [0,255]. The location of the pels is determined from mb_row, mb_column and the pattern_code list.

3.3.3 Predictive-coded macroblocks in P2- and B-pictures

The predictive-coded macroblocks in P2- and B-Pictures are decoded in four steps.

First, the value of a forward motion vector for a macroblock is reconstructed from the retrieved forward motion vector information, and the forward motion vector reconstructed for the previous macroblock. However, for P2- and B-coded pictures, the previous reconstructed motion vectors are reset only for the first macroblock in a slice and when the last decoded macroblock was an intra-coded macroblock. If no forward motion vector data exists for the current macroblock, the motion vectors are obtained by

```
    recon_right_for[] = recon_right_for_prev[],
    recon_down_for[] = recon_down_for_prev[].
```

Second, the value of the backward motion vector for the macroblock is reconstructed from the retrieved backward motion vector information and the backward motion vector reconstructed for the previous macroblock using the same procedure as for calculating the forward motion vector in P2- and B-pictures. In the case of P2-picture decoding, the forward motion vector is used for the far picture, and the backward motion vector is used for the near picture.

The following variables result from applying the algorithm in clause 3.3.2, modified as described in the previous two paragraphs:

```
right_for[], right_half_for[], down_for[], down_half_for[]
right_back[], right_half_back[], down_back[], down_half_back[]
```

which define the integral and half pel value of the rightward and downward components of the forward motion vector (which references the far picture in P2-picture and the past picture in B-picture) and the backward motion vector (which references the near picture in P2-picture and the future picture in B-picture).

Third, the pels of the decoded picture are calculated. If only forward motion vector information was retrieved for the macroblock, then pel_new[][] of the decoded picture is calculated according to the formulas in the predictive-coded macroblock section. If only backward motion vector information was retrieved for the macroblock, then pel_new[][] of the decoded picture is calculated according to the formulas in the predictive-coded macroblock section, with "back" replacing "for", and "pel_future[][][]" replacing "pel_past[][][]". If both forward and backward motion vectors information are retrieved, then let pel_new_for[][] be the value calculated from the past picture by using the reconstructed forward motion vector, and let pel_new_back[][] be the value calculated from the future picture by using the reconstructed backward motion vector. Then the value of pel_new[][] is calculated by:

```
pel_new[][] = ( pel_new_for[][] + pel_new_back[][] ) // 2 ;
```

Fourth, the DCT coefficients for each block present in the macroblock are reconstructed by:

```
for ( m=0; m<8; m++ ) {
        for ( n=0; n<8; n++ ) {
                i = scan[][m][n] ;
                dct_recon[m][n] = ( 2 * dct_zz[i] * ( 16 + quantizer_scale
                                                   * non_intra_quant[m][n] ) ) / 16 ;
                if ( ( dct_recon[m][n] & 1 ) == 0 )
                        dct_recon[m][n] = dct_recon[m][n] - Sign(dct_recon[m][n])
;
                if (dct_recon[m][n] > 2047) dct_recon[m][n] = 2047 ;
                if (dct_recon[m][n] < -2048) dct_recon[m][n] = -2048 ;
                if ( dct_zz[i] == 0 )
                        dct_recon[m][n] = 0 ;
        }
}
```

If the reconstructed block is a luminance block, then the first index of scan[][][] shall be 2. If the reconstructed block is a chrominance block, then the first index of scan[][][] shall be 3.

NOTE - dct_recon[m][n] = 0 (for all m, n) in skipped macroblocks and when pattern[i] == 0.

Once the dct coefficients are reconstructed, the inverse DCT transform is applied to obtain the inverse transformed pel values in the range [-256, 255]. The inverse DCT pel values are added to pel_new[][], which were computed above from the motion vectors, with the result of the addition being limited to the interval [0,255]. The location of the pels is determined from mb_row, mb_column and the pattern_code list.

3.3.4 Skipped macroblocks

Some macroblocks are not stored, i.e. neither motion vector information nor DCT information is available to the decoder. These macroblocks occur when the macroblock_address_increment is greater than 1. The macroblocks for which no data is stored are called "skipped macroblocks".

In I-pictures, all macroblocks are coded and there are no skipped macroblocks.

In P0- and P1-pictures, the skipped macroblock is defined to be a macroblock with a reconstructed motion vector equal to zero and no DCT coefficients.

In P2- and B-pictures, the skipped macroblock is defined to have the same macroblock_type (forward, backward, or both motion vectors) as the prior macroblock, differential motion vectors (motion_horizontal/vertical_forward/backward) equal to zero, the same interpolation mode and differential motion vector(s), and no DCT coefficients. In P2- and B-pictures, a skipped macroblock shall not follow an intra-coded macroblock.

4. Status of the proposed algorithm

4.1 Compatibility

This algorithm cannot afford to give backward compatibility to MPEG1 and H.261. As for forward compatibility, it can give compatible codec in the form of a switchable encoder / decoder. Most part of the encoder / decoder for the two standards can be commonly used, and only the predictor, VLC tables, and coding controller are to be switched.

4.2 Random access

Random access is realized by accessing first to the nearest entry point (= I-picture) before the field to be accessed from all of the periodically inserted entry points. In this proposal, 1 GOP is constructed from 24 fields. 4 fields of I or P-pictures are necessary to decode B-picture in some cases. So, the longest path for random accessing is the access to the set of B2 and B3-picture located just before the entry point in input order and the accessing path is shown below.

$$(I) \to (P0) \to (B0) \to (B1) \to (B2) \to (B3) \to (P1) \to (P2) \to (B0) \to (B1) \to (B2) \to (B3) \to (P1) \to (P2) \to (B0) \to (B1) \to (B2) \to (B3) \to \textcircled{P1} \to \textcircled{P2} \to (B0) \to (B1) \to (B2) \to (B3) \to \textcircled{I} \to \textcircled{P0} \to (B0) \to (B1) \to B2^{*} \to B3^{*}$$

($\bigcirc$ denotes the field used for predicting the attended B2* and B3*,
and ( ) denotes a discarded field not used for prediction)

In this case, it is necessary to read out the amount of data corresponding to 1 GOP + I + P0 + B0 + B1 + B2 + B3. In this proposal, the amount of data is limited to be within 50 % of the target amount of data for 1 GOP. So, the maximum delay time becomes the time to read out the data corresponding to 1.5 GOP + 4 B, i.e about $24 / 60 \times 1.5 + \text{alpha} \approx 0.6$ sec.

Actually, there can be some fluctuation in the data amount for rate control for each GOP and the maximum delay becomes slightly more. The maximum value of the control fluctuation observed from actual simulations using 8 sequences was 0.62 sec.

4.3 Fast forward / fast reverse

Rate control is executed in this proposal so that the amount of generated information for a set of I-picture + P0-picture, which is used for fast forward and fast reverse playing, don't exceed 50 % of the target amount of data allocated to 1 GOP. This means that a set of I and P0-picture can be read out within 0.2 sec when working in a fast playing mode. For example, 6 times multiple speed playing mode is realized by decoding a set of I and P0-picture and skipping two sets every 0.2 sec shown below (each number denotes the continuous picture number).

```
normal play    0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20...
6 times fast   0 1 0 1 0 1 0 1 0 1  0  1 72 73 72 73 72 73 72 73 72..
```

So, delay time for fast forward and fast reverse playing is 0.2 sec.

4.4 Coding / decoding delay

Coding / decoding delay is derived mainly from the delay caused by reordering the coding / decoding order owing to bi-directional prediction and the delay caused by the input / output buffer. The delay in format transformation at input / output, calculation at the filters or DCT is ignorablly small compared to these two.

First, the delay by reordering is investigated below. The input order of the fields at the encoder, the coded order of the fields at the encoder = the output order of the fields at the encoder = the input order of the fields at the decoder, the output order of the fields at the decoder is shown in Fig. 4-1. As shown in the figure, this delay corresponds to 6 fields at the decoder for the I and P-pictures and 6 fields at the encoder for the B-picture at the shortest even under the condition that the data of each field continues without any interval to make the delay short. (Note that this delay becomes 4 fields at the encoder and 2 fields at the decoder for I and P0-picture at the start of the sequence. Even in this case, the sum of the delay is the same.) Adding to this, still one field delay is necessary to read the activity of the I-picture before coding. The conclusion is that the coding / decoding delay for each picture corresponds to 7 fields, i.e. 7 / 60 = 0.117 sec.

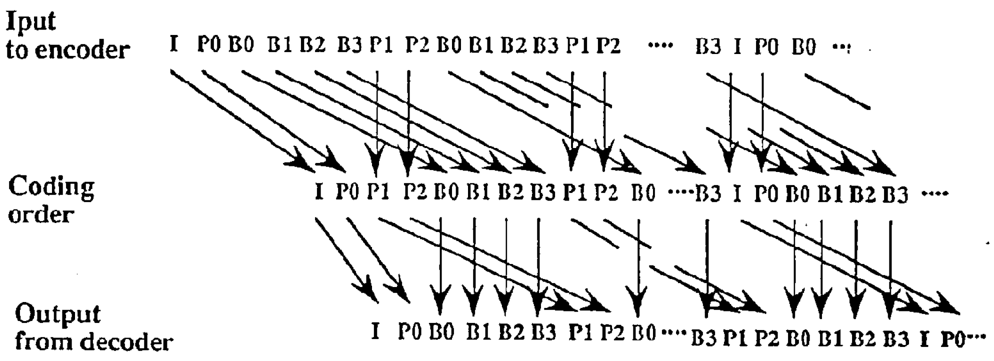

Iput to encoder
I P0 B0 B1 B2 B3 P1 P2 B0 B1 B2 B3 P1 P2 ···· B3 I P0 B0 ···
Coding order
I P0 P1 P2 B0 B1 B2 B3 P1 P2 B0 ····B3 I P0 B0 B1 B2 B3 ····
Output from decoder
I P0 B0 B1 B2 B3 P1 P2 B0····B3 P1 P2 B0 B1 B2 B3 I P0···

Fig. 4-1 Delay caused by reordering at encoder and decoder

Next, the delay caused by the buffer is investigated. Suppose the receiving data buffer of the decoder is modeled by a fixed rate input and instantaneous output of data when data are needed to be decoded, and the output data buffer is modeled by an instantaneous input when data are generated and at a fixed output rate. When rate control is executed so as not to make any overflow and underflow at the transmitting buffer, it is also warranted that the receiving buffer never overflows or underflows by preparing a buffer of the same capacity at the receiver. In this case, the total delay at the transmitting buffer and the receiving buffer is the capacity of the transmitting buffer (= the capacity of receiving buffer) divided by the output rate.

In this proposal, the amount of transmitting buffer and receiving buffer is determined to be 2^{21} (≈ 2M) bits at 9 Mbps. (As for 4 Mbps, $2^{11} \times 4/9$ Mbits are used for buffer). So, the delay by the buffer is ≈ 0.233 sec.

Considering the above two main factors, the coding / decoding delay by this proposal is estimated to be about 350 msec.

5. Implementation

5.1 List of picture buffer and data buffer

1) Encoder

Picture/data buffer	Number	Size(bits)
Picture buffer for Y	15	20736000 (720×240×8×15)
Picture buffer for C	9	12441600 (720×240×8×9)
Output data buffer	1	2097152 (2^{21})

2) Decoder

Picture/data buffer	Number	Size(bits)
Receiving data buffer	1	2097152 (2^{21})
Picture buffer for Y	4	5529600 (720×240×8×4)
Picture buffer for C	4	5529600 (720×240×8×4)

5.2 List for each module with functional description

5.2.1 Encoder

1) Input field memory

Memory	Size(words)	Width(bits)	On/Off chip
Input picture memory(Y 10 fields)	1728000	8	Off
Input picture memory(C 4 fields)	691200	8	Off
Picture delay memory(Y,C 1 field)	345600	8	Off
Block line memory(Y,C)	46080	8	On

Figure 5-1 shows a block diagram for the input field memory. For both Y and C signals, B0, B1, B2 and B3 pictures are stored in the input picture memories for changing the order of pictures. For the Y signal, I, Po, P1 and P2 pictures are also stored in the input picture memories. Stored pictures for the Y signal are used for the first stage motion vector search in the motion vector estimator 1. Y and C signals are multiplexed after scan format conversion by the block line memories. Then, they are delayed by one field for intra activity calculation.

2) Motion vector estimator 1

Memory	Size(words)	Width(bits)	On/Off chip
Motion vector memory	16200	7+6	On
Search data register in MVD1	8096	8	On
Distortion register in MVD1	1860	15	On
Register in acc. dist.calc. MVD1	1860	15	On

Addition	Width(bits)	Operations/sec
Subtraction in dist.calc. MVD1	8	16070400000
Addition in acc. dist.calc. MVD1	15	16070400000
Vector addition in MVD	7	540000

Figure 5-2 shows a block diagram for the motion vector estimator 1.

In this module, motion vectors for the first stage are calculated by the so called telescopic search, using the pictures before coding stored in the input field memory.

For each step in the telescopic search, four motion vector detectors (MVD1a..d) calculate intermediate motion vectors between two pictures, whose separation distance is one field or two, as follows. At first, search data are stored in the search data registers in the MVD1, by reading data in the input field memory addressed by the start motion vector, which has been stored in the motion vector (MV) memory at the previous search stage. Then, 31×15 distortion calculators derive the distortion values for all offset vector candidates in parallel. Finally, the motion vector is updated by adding the start motion vector and offset vector determined in the distortion comparator, and stored in the MV memory.

The picture to be referenced as an origin and the picture to be searched for in motion vector detection, in each step for each MVD, are illustrated in Table 5-1. The same pictures must be stored in two different field memories, occasionally, because data in the same picture with a different address must be read by different MVDs.

3) Motion vector estimator 2 with adaptive predictor

Memory	Size(words)	Width(bits)	On/Off chip
Search data register in MVD21	3840	8	On
Distortion register in MVD21	36	15	On
Register in acc. dist.calc.1 MVD21	36	15	On
Search data register in MVD22	11928	8	On
Distortion register in MVD22	72	15	On
Register in acc. dist.calc.2 MVD22	72	15	On
Temporary data register	6664	8	On
Macroblock delay	1024	8	On

Addition	Width(bits)	Operations/sec
Subtraction in dist.calc.1 MVD21	8	373248000

Addition in acc. dist.calc.1 MVD21	15	373248000
Vector addition in MVD21	7	648000
Addition in field interpolation filter	8	440316000
Subtraction in dist.calc.2 MVD22	8	746496000
Addition in acc. dist.calc.2 MVD22	15	746496000
Subtraction for r2 in MVD22	8	41472000
Addition for frame interpolation mode signal	8	41472000
Addition for bi-directional prediction signal	8	20736000
Subtraction in dist.calc.3	8	20736000
Addition in acc. dist.calc.3	15	20736000

Figure 5-5 shows a block diagram for motion vector estimator 2 with the adaptive predictor.

In this module, an adaptive inter field / inter frame prediction signal is generated from the decoded pictures, by the second stage motion vector estimation, as follows.

At first, for each decoded picture, motion vector detector 21(MVD21) refines the first stage motion vector, by searching for the decoded picture around a point within the ±1 range.

Second, the motion vector detector 22(MVD22) searches within the range of a half-pixel for both the field and frame interpolation mode, as described in Section 2. Search for the field interpolation mode is implemented, using the field interpolated picture, generated by the field interpolation filter. For the frame interpolation mode, search is accomplished using both the field interpolated picture, and the picture of the other field, pointed out by the refined motion vector detected by the MVD21.

Third, distortion comparator 1 selects the optimum interpolation, and the optimum prediction signal for forward or backward direction is generated.

Finally, forward, backward and interpolative prediction signals are compared, and the optimum one is selected.

5) Subtracter for prediction error signal

Addition	Width(bits)	Operations/sec
Subtraction for prediction error signal	8	20736000

6) Activity calculator

Memory	Size(words)	Width(bits)	On/Off chip
Block delay	64	8	On
Slice activity register	150	22	On

Addition	Width(bits)	Operations/sec
Addition in block DC calculator	14	864000
DC Subtraction	8	864000
Addition in slice activity calculator	22	20736000
Addition in picture activity calculator	27	1800

Figure 5-10 shows a block diagram for the activity calculator. For I pictures, absolute differences between pel values and the average value of the block, calculated by the block DC calculator, are derived as a measure of the activity. For P or B pictures, the absolute values for the AC coefficients of the result of DCT are used. Their sum for every slice and for one picture are calculated by the slice and picture activity accumulators.

7) DCT

Memory	Size(words)	Width(bits)	On/Off chip
Register in 8-point processor	80	16	On
Matrix transposition RAM	64	16	On
Scan conversion RAM	128	16	On

Addition	Width(bits)	Operations/sec
Addition in 8-point processor	16	165888000

Multiplication	Width(bits)	Operations/sec
Multiplication in 8-point processor	16	82944000

Table	Size(words)	Width(bits)	Fix/Dwn	Look up/sec
Scan table	256	6	Dwn	20736000
Coef table	4	16+16	Fix	20736000

Figure 5-11 shows the DCT block diagram. In this DCT module, a fast calculating algorithm described in Reference 3, whose calculation signal flow is depicted in Fig. 5-13, is used.
It consists of two 8-point processors for one-dimensional transform with a coefficient table, two 8 word buffers, matrix transposition RAMs and scan conversion RAMs with a scan table.

Each 8-point processor consists of three 8 word buffers, three adding stages and two multiplying stages. Addition or multiplication is carried out at pixel rate according to the signal flow graph shown in Fig. 5-13.

8) Quantizer

Multiplication	Width(bits)	Operations/sec
Multiplication in quantizer	12	20736000

Table	Size(words)	Width(bits)	Fix/Dwn	Lookup/sec
Quantize^{-1} table	4096	12	Dwn	20736000

Figure 5-14 shows the quantizer block diagram. The inverse values for quantization steps are stored in the quantize^{-1} table. Quantization is implemented by multiplying the selected value in this table, to the absolute value for the DCT coefficient.

9) Dequantizer

Addition	Width(bits)	Operations/sec
Offset addition	12	20736000

Multiplication	Width(bits)	Operations/sec
Q_scale multiplication	5×8	20736000
DCT coeff multiplication	8×12	20736000

Table	Size(words)	Width(bits)	Fix/Dwn	Lookup/sec
Quantize table	128	8	Dwn	20736000

Figure 5-15 shows the dequantizer block diagram. The selected value in the quantize table and the quantizer scale are multiplied. The quantization step size is derived by adding the offset value and the multiplied value. Dequantization is carried out by multiplying the quantization step size by the quantized DCT coefficient.

10) IDCT

Memory	Size(words)	Width(bits)	On/Off chip
Register in 8-point processor	80	16	On
Matrix transposition RAM	64	16	On
Scan conversion RAM	128	16	On

Addition	Width(bits)	Operations/sec
Addition in 8-point processor	16	165888000

Multiplication	Width(bits)	Operations/sec
Multiplication in 8-point processor	16	82944000

Table	Size(words)	Width(bits)	Fix/Dwn	Lookup/sec
Scan table	256	6	Dwn	20736000
Coef table	4	16+16	Fix	20736000

Figure 5-16 shows the IDCT block diagram. The operation for each stage in 8-point processors is depicted in Fig. 5-18.

11) Adder for local decoded signal

Addition	Width(bits)	Operations/sec
Addition for local decoded signal	8	20736000

12) Local decoded field memory

Memory	Size(words)	Width(bits)	On/Off chip
Decoded picture memory(Y,C 4 field)	1382400	8	Off

13) Encoder output stage (VLC, MUX and output data buffer)

Memory	Size(words)	Width(bits)	On/Off chip
Output data buffer	65536	32	Off

Addition	Width(bits)	Operations/sec
Subtraction in macroblock addr. DPCM encoder	6	81000(max)
Subtraction in motion vector DPCM encoder	8	648000(max)
Subtraction in block DC DPCM encoder	9	324000(max)

Table	Size(words)	Width(bits)	Fix/Dwn	Lookup/sec
VLC table for M.B._address	64	12	Dwn	81000(max)
VLC table for M.B._type	64	7	Dwn	81000(max)
VLC table for C._B._P	16	9	Dwn	81000(max)
VLC table for motion vector	16	11	Dwn	648000(max)
VLC table for dct_dc_size	32	9	Dwn	324000(max)
VLC table for dct_coef	8192	17	Dwn	20736000(max)

Figure 5-19 shows a block diagram for the encoder output stage. Data in the sequence/GOP/picture/slice layer are generated by the header generator. Data in the macroblock layer are generated by the macroblock data generator, which contains the macroblock address DPCM encoder, motion vector DPCM encoder and VLC Table 1, using the motion vector data from motion vector estimator 2. Data in the block layer are generated by the block data generator, which contains the block DC DPCM encoder, the zero run counter and VLC Table 2, using the quantized DCT coefficient value from the quantizer. Data in each layer are multiplexed under the sequencer control, and are stored in the output data buffer. The rate controller determines the Q_scale value, using the activity data and the status for the output data buffer.

5.2.2 Decoder

1) Decoder input stage (receiving data buffer, VLD and DMUX)

Memory	Size(words)	Width(bits)	On/Off chip
Receiving data buffer	65536	32	Off
Macroblock mode memory	90	4	On
Motion vector memory	360	8+7	On
Q_scale memory	90	5	On
DCT coef. memory	512	9	On

Addition	Width(bits)	Operations/sec
Addition in macroblock addr. DPCM decoder	6	81000(max)
Addition in motion vector DPCM decoder	8	648000(max)
Addition in block DC DPCM decoder	9	324000(max)
Addition in run to position translator	6	20736000(max)

Table	Size(words)	Width(bits)	Fix/Dwn	Lookup/sec
VLC decode table	8192	20	Dwn	27×10^6

Figure 5-20 shows a block diagram of the decoder input stage. Data from the receiving buffer are variable length decoded, and are demupltiplexed into macroblock data, block data and other data in sequence/GOP/picture layer. The macroblock data are decoded by the macro block data decoder, which contains the macroblock address decoder and the motion vector decoder. The block data are decoded by the block data decoder, which contains the block DC DPCM decoder and the translator from run to position. These decoded values are stored into the slice data memory.

A detailed block diagram of the VLD and DMUX is shown in Fig.5-21. The sequencer determines the kind of VLC/FLC to be decoded, by the start code detector output and the contents/inputs of the register array. This VLD can decode one variable length code per one clock. One variable length code is divided into three parts, namely, the front zero/one runs and the front_next_code and the rear_code. The length of zero/one run is detected by the zero/one run detector. This value and the front_next_code are fed into the VLC decode table. The decoded value for the front_code and the number of bits for the rear_code are generated by the table. The rear_code is decoded by the rear code detector. The clock rate is the same as that for the pel rate (27MHz). The timing periods for decoding the data in the slice/macroblock layer and the sequence/GOP/picture layer can be maintained by the horizontal and vertical blanking periods.

2) Adaptive predictor

Addition	Width(bits)	Operations/sec

Addition in field interpolation filter	8	181440000
Addition for frame interpolation mode signal	8	41472000
Addition for bi-directional prediction signal	8	20736000

Figure 5-22 shows a block diagram of the adaptive predictor for the decoder. The function for this module is the same as that for the adaptive predictor in the encoder.

3) Dequantizer

Addition	Width(bits)	Operations/sec
Offset addition	12	20736000

Multiplication	Width(bits)	Operations/sec
Q_scale multiplication	5×8	20736000
DCT coeff multiplication	8×12	20736000

Table	Size(words)	Width(bits)	Fix/Dwn	Lookup/sec
Quantize table	128	8	Dwn	20736000

This module is the same as that for the encoder.

4) IDCT

Memory	Size(words)	Width(bits)	On/Off chip
Register in 8-point processor	80	16	On
Matrix transposition RAM	64	16	On
Scan conversion RAM	128	16	On

Addition	Width(bits)	Operations/sec
Addition in 8-point processor	16	165888000

Multiplication	Width(bits)	Operations/sec
Multiplication in 8-point processor	16	82944000

Table	Size(words)	Width(bits)	Fix/Dwn	Lookup/sec
Scan table	256	6	Dwn	20736000
Coef table	4	16+16	Fix	20736000

This module is the same as that for the encoder.

5) Adder for decoded signal

Addition	Width(bits)	Operations/sec
Addition for decoded signal	8	20736000

6) Decoded field memory with output

Memory	Size(words)	Width(bits)	On/Off chip
Decoded picture memory(Y,C 4 field)	1382400	8	Off
Block line memory(Y,C)	23040	8	On

Figure 5-23 shows a block diagram for the decoded field memory with output. Decoded B pictures are directly fed into the block line memory, which is used for scan conversion to output signals. I and P pictures are stored in the decoded picture memory, and they are used as inputs for both the adaptive predictor and the block line memory.

5.3 Supplemental document in implementation

This is the supplemental document explaining how the numbers in the lists in Section 5.2 are calculated.

5.3.1 Encoder

1) Input field memory

Input picture memory(Y 10 fields)	1728000 = #pel_field × 10
Input picture memory(Y 4 fields)	691200 = #pel_field × 4
Picture delay memory(Y,C 1 fields)	345600 = #pel_field × 2
Block line memory	46080 = #h_field × #v_macro × 4 × 2
	4: Number of block lines
	2: Number of block line memories

2) Motion vector estimator 1

Motion vector memory 16200 = #macro_field × 12
12: Number of motion vector memories

Search data register in MVD1

8096 = (#h_macro + 2 × #h_search) ×
(#v_macro + 2 × #v_search) × 2 × #MVD1

2: Double buffer

#MVD1 = 4 :Number of MVD1s

Distortion register in MVD1

Register in acc. dist.calc. MVD1

$1860 = \#search \times \#MVD1$

Subtraction in dist.calc. MVD1

Addition in acc. dist.calc MVD1

$16070400000 = \#search \times \#pel\_field \times \#field\_sec \times \#MVD1 \times \#MD1\_act$

$\#MVD1\_act = (\#MVD1 \times \#field\_gop - (2+2+4+8)) / \#MVD1 / \#field\_gop$

: Average operating ratio for MVD1s

(MVD1s do not operate all the time.)

Vector addition in MVD

$540000 = \#macro\_field \times \#field\_sec \times \#MVD1 \times \#MVD1\_act \times 2$

2: Horizontal and vertical vector

3) Motion vector estimator 2 with adaptive predictor

Search data register in MVD21

$3840 = ((\#h\_macro + 4) \times (\#v\_macro + 4) + (\#h\_macro / 2 + 2) \times (\#v\_macro + 4) \times 2) \times 2 \times \#MVD21$

2: Double buffer

#MVD21 = 4: Number of MD21

Distortion register in MVD21

Register in acc. dist.calc. MVD21

$36 = \#search\_21 \times \#MVD21$

#search_21 = 9: Number of searching points in one MVD21

Search data register in MD22

$11928 = (((\#h\_macro + 3) \times 2 - 1) \times ((\#v\_macro + 3) \times 2 - 1) +$:Y

$((\#h\_macro / 2 + 1) \times 2 - 1) \times ((\#v\_macro + 3) \times 2 - 1) \times 2$:C×2

$) \times 2 \times \#MVD22$

2: Double buffer

#MVD22 = 4 :Number of MVD22

Distortion register in MVD22

Register in acc. dist.calc. MVD22

$72 = \#search\_22 \times \#MVD22$

#search_22 = 18 : Number of searching points in one MVD22

Temporary data register

$6664 = ((\#h\_macro \times 2 + 1) \times (\#v\_macro \times 2 + 1) +$:Y

$(\#h\_macro / 2) \times (\#v\_macro \times 2 + 1) \times 2$:C×2

$) \times 2 \times \#T.d.r$

2: Double buffer

#T.d.r: Number of temporary data registers

Macroblock delay

$1024 = \#pel\_macro \times 8$

8: two macroblock(Y) and three macroblock(Y,C)

Subtraction in dist.calc. MVD21

Addition in acc. dist.calc MVD21

$373248000 = \#search\_21 \times \#pel\_field \times \#field\_sec \times \#MVD21$

Vector addition in MVD21

$648000 = \#macro\_field \times \#field\_sec \times \#MVD21 \times 2$

2: Horizontal and vertical vector

Addition in field interpolation filter

$440316000 = \#a.f.i.f\_macro \times \#macro\_field \times field\_sec \times \#f.i.f$

$$\begin{aligned}\#a.f.i.f\_macro = (\ & (\#v\_macro + 3) \times (\#h\_macro + 4) + && \text{Ver. Y}\\ & (\#h\_macro + 3) \times ((\#v\_macro + 4) \times 2 - 1) + && \text{Hor. Y}\\ & ((\#v\_macro + 3) \times (\#h\_macro / 2 + 3) + && \text{Ver. C}\\ & (\#h\_macro / 2 + 2) \times ((\#v\_macro + 4) \times 2 - 1) && \text{Hor. C}\\ &) \times 2\ ) = 1359 &&\end{aligned}$$

: Addition in one field interpolation filter per macroblock

$\#f.i.f = 4$: Number of field interpolation filter

Subtraction in dist.calc. MVD22

Addition in acc. dist.calc MVD22

$746496000 = \#search\_22 \times \#pel\_field \times \#field\_sec \times \#MVD22$

Subtraction for r2 in MVD22

$41472000 = \#pel\_field \times \#field\_sec \times \#MVD22$

Addition for frame interpolation mode signal

$41472000 = \#pel\_field \times 2 \times \#field\_sec \times 2$

2: Y and C

2: Forward and backward

Addition for bi-directional prediction signal

$20736000 = \#pel\_field \times 2 \times \#field\_sec \times 2$

2: Y and C

Subtraction in dist.calc.3

Addition in acc. dist.calc 3

$20736000 = \#pel\_field \times 2 \times \#field\_sec \times 2$

2: Two candidates

5) Subtracter for prediction error signal

$20736000 = \#pel\_field \times 2 \times \#field\_sec$

2: Y and C

6) Activity calculator

Slice activity register

$150 = \#v\_field / \#v\_macro \times 5$

5: I, P0, P1, P2 and B

Addition in block DC calculator

$864000 = \#pel\_field \times 2 \times \#field\_sec \times 1/24$

1/24: Number of I pictures in GOP

Addition in slice activity calculator

Addition in picture activity calculator

$1800 = \#slice\_field \times \#field\_sec$

7) DCT

Addition in 8-point processor

165888000 = #pel_field × 2 × #field_sec × (3 + 2 × 1/2) × 2

3: Number of add. stages in one 8-point processor

2: Number of mult. stages in one 8-point processor

2: Number of 8-point processors

Multiplication in 8-point processor

82944000 = #pel_field × 2 × #field_sec × 4

4: Number of mult. stages in two 8-point processors

Scan table, Coef table

20736000 = #pel_field × 2 × #field_sec

8) Quantizer

Multiplication in quantizer, Quantize^{-1} table

20736000 = #pel_field × 2 × #field_sec

9) Dequantizer

Offset addition, Q_scale multiplication, DCT coeff multiplication

20736000 = #pel_field × 2 × #field_sec

10) IDCT

same as DCT

11) Adder for local decoded signal

20736000 = #pel_field × 2 × #field_sec

2: Y and C

12) Local decoded field memory

Decoded picture memory(Y,C 4 fields) 1382400 = #pel_field × 2 × 4

13) VLC, MUX and output data buffer

Subtraction in macroblock addr. DPCM enc.	81000 = #macro_field × #field_sec
Subtraction in MV DPCM encoder	648000 = #macro_field × 2 × 4 × #field_sec 2: Horizontal and vertical 4: Four reference fields
Subtraction in block DC DPCM encoder	324000 = #macro_field × 4 × #field_sec 4: Four blocks in one macroblock

VLC table for M.B._address

VLC table for M.B._type

VLC table for C._B._P.	81000 = #macro_field × #field_sec
VLC table for motion vector	648000 = #macro_field × 2 × 4 × #field_sec 2: Horizontal and vertical 4: Four reference fields
VLC table for dct_dc_size	324000 = #macro_field × 4 × #field_sec 4: Four blocks in one macroblock
VLC table for dct_coef	20736000 = #pel_field × 2 × #field_sec 2: Y and C

5.3.2 Decoder

1) Receiving data Buffer, VLD and DMUX

Macroblock mode memory	90 = #macro_slice × 2 2: Double buffer
Motion vector memory	360 = #macro_slice × 4 × 2 4: Four reference fields 2: Double buffer
Q_scale memory	90 = #macro_slice × 2 2: Double buffer
DCT coef. memory	512 = #pel_macro × 2 × 2 2: Y and C 2: Double buffer

Addition in macroblock addr. DPCM dec.	81000 = #macro_field × #field_sec
Addition in motion vector DPCM dec.	648000 = #macro_field × 2 × 4 × #field_sec 2: Horizontal and vertical 4: Four reference field
Addition in block DC DPCM decoder	324000 = #macro_field × 4 × #field_sec 4: Four blocks in one macroblock
Addition in run to position translator	20736000 = #pel_field × 2 × #field_sec
VLC decode table	8192 = $2^{(4+4+5)}$ 4: Number of bits for VLC code selection 4: Number of bits for 0/1 run number 5: Number of bits for front next_code 20 = 3 + 4 + 1 + 12 3: Number of bits for front_next_code length 4: Number of bits for rear_code length 1: Escape flag 12: Number of bits for Decode value

3) Adaptive predictor

Addition in field interpolation filter

181440000 = #a.f.i.f(d)_macro × #macro_field × field_sec × #f.i.f

#a.f.i.f_macro =	(#h_macro + 2) × #v_macro +	Ver. Y
	#h_macro × #v_macro +	Hor. Y

$((\#h\_macro / 2 + 2) \times \#v\_macro +$	Ver. C
$\#h\_macro / 2 \times \#v\_macro)$	Hor. C
$) \times 2 = 560$	

: Addition in one field interpolation filter per one macro block

$\#f.i.f = 4$: Number of field interpolation filters

Addition for frame interpolation mode signal

$41472000 = \#pel\_field \times 2 \times \#field\_sec \times 2$

2: Y and C

2: Forward and backward

Addition for bi-directional prediction signal

$20736000 = \#pel\_field \times 2 \times \#field\_sec$

2: Y and C

4) Dequantizer

Same as that for the encoder

5) IDCT

Same as that for the encoder

6) Adder for decoded signal

Same as the adder for the local decoded signal in the encoder

7) Decoded field memory with output

Decoded picture memory(Y,C 4 field)	$1382400 = \#pel\_field \times 2 \times 4$
Block line memory	$23040 = \#h\_field \times \#v\_macro \times 2 \times 2$
	2: Y and C
	2: Double buffer

$\#pel\_field = \#h\_size \times \#v\_size$	:Number of pels per field (Y or C)
$\#h\_field = 720$	:Horizontal size of a field memory (Y)
$\#v\_field = 240$	:Vertical size of a field memory (Y or C)
$\#h\_macro = 16$	:Horizontal size of a macro block (Y)
$\#v\_macro = 8$	:Vertical size of a macro block (Y or C)
$\#pel\_macro = \#h\_macro \times \#v\_macro$	:Number of pels per macro block (Y)
$\#macro\_field = \#pel\_field / \#pel\_macro$	:Number of macro blocks per field
$\#macro\_slice = \#h\_field / \#h\_macro$	:Number of macro blocks per slice
$\#slice\_field = \#v\_field / \#v\_macro$	:Number of slices per field
$\#h\_search = 15$	:Horizontal search region (+/-) in the MVD1
$\#V\_search = 7$	:Vertical search region (+/-) in the MVD1
$\#search = (1 + 2 \times \#h\_search) \times (1 + 2 \times \#v\_search)$	:Number of search points in the MVD1
$\#field\_sec = 60$	:Number of fields per sec
$\#field\_gop = 24$	:Number of fields per GOP

Input order		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
Input picture		B0a	B1a	B2a	B3a	Ia	P0a	B0b	B1b	B2b	B3b	P1b	P2b	B0c	B1c	B2c	B3c	P0c	P1c	B0d	B1d	B2d	B3d	P0d	P1d	B0a	B1a	B2a	B3a	Ia	P0a	B0b	B1b	B2b	B3b
MVD 1a	Referenced	P0d	P0d	P0d	P0d			Ia	Ia	Ia	Ia	Ia	Ia	P1b	P1b	P1b	P1b	P1b	P1b	P0c	P0c	P0c	P0c	P0c	P0c	P0d	P0d	P0d	P0d			Ia	Ia	Ia	Ia
	Searched	B0a	B1a	B2a	B3a			B0b	B1b	B2b	B3b	P1b	P2b	B0c	B1c	B2c	B3c	P0c	P1c	B0d	B1d	B2d	B3d	P0d	P1d	B0a	B1a	B2a	B3a			B0b	B1b	B2b	B3b
MVD 1b	Referenced	P1d	P1d	P1d	P1d			P0a	P0a	P0a	P0a	P0a	P0a	P2b	P2b	P2b	P2b	P2b	P2b	P1c	P1c	P1c	P1c	P1c	P1c	P1d	P1d	P1d	P1d			P0a	P0a	P0a	P0a
	Searched	B0a	B1a	B2a	B3a			B0b	B1b	B2b	B3b	P1b	P2b	B0c	B1c	B2c	B3c	P0c	P1c	B0d	B1d	B2d	B3d	P0d	P1d	B0a	B1a	B2a	B3a			B0b	B1b	B2b	B3b
MVD 1c	Referenced	P0d	P0d	P0d		Ia	Ia	Ia	Ia	Ia		P1b	P1b	P1b	P1b	P1b		P0c	P0c	P0c	P0c	P0c		P0d	P0d	P0d	P0d	P0d		Ia	Ia	Ia	Ia	Ia	
	Searched	B2d	B0d	B1d		B3a	P0a	B2a	B0a	B1a		B3b	P2b	B2b	B0b	B1b		B3c	P1c	B2c	B0c	B1c		B3d	P1d	B2d	B0d	B1d		B3a	P0a	B2a	B0a	B1a	
MVD 1d	Referenced	P1d	P1d	P1d			P0a	P0a	P0a	P0a			P2b	P2b	P2b	P2b			P1c	P1c	P1c	P1c			P1d	P1d	P1d	P1d			P0a	P0a	P0a	P0a	
	Searched	B1d	B0d	B2d			B3a	B1a	B0a	B2a			B3b	B1b	B0b	B2b			B3c	B1c	B0c	B2c			B3d	B1d	B0d	B2d			B3a	B1a	B0a	B2a	
Coding picture		P1d	B0d	B1d	B2d	B3d	Ia	P0a	B0a	B1a	B2a	B3a	P1b	P2b	B0b	B1b	B2b	B3b	P0c	P1c	B0c	B1c	B2c	B3c	P0d	P1d	B0d	B1d	B2d	B3d	Ia	P0a	B0a	B1a	B2a

Table 5-1 Timing table of motion vector estimator 1

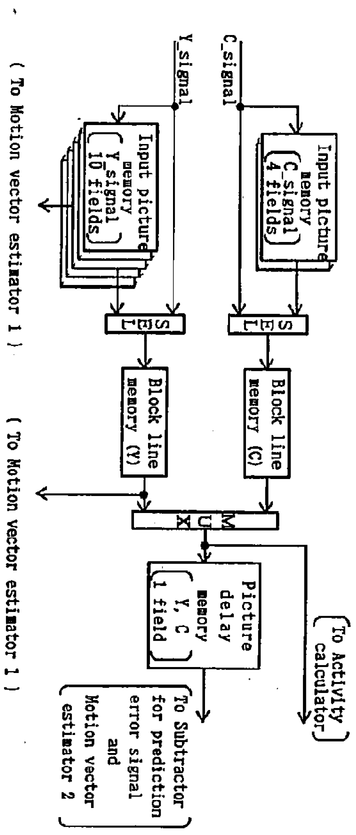
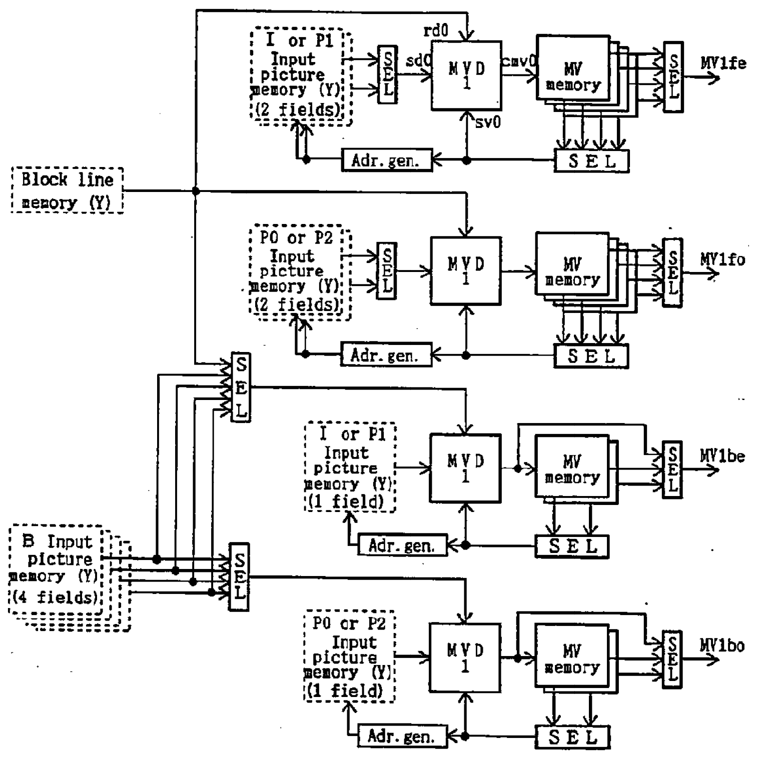

Y_signal
C_signal
Input picture memory (Y_signal 10 fields)
Input picture memory (C-signal 4 fields)
SEL
SEL
Block line memory (Y)
Block line memory (C)
MUX
Picture delay memory (Y, C 1 field)
(To Motion vector estimator 1)
(To Motion vector estimator 1)
(To Activity calculator)
(To Subtractor for prediction error signal and Motion vector estimator 2)

Figure 5-1 Input field memory


I or P1 Input picture memory (Y) (2 fields)
rd0
sd0
S E L
M V D 1
cmv0
MV memory
MV1fe
sv0
Adr. gen.
S E L
Block line memory (Y)
P0 or P2 Input picture memory (Y) (2 fields)
MV1fo
I or P1 Input picture memory (Y) (1 field)
MV1be
B Input picture memory (Y) (4 fields)
P0 or P2 Input picture memory (Y) (1 field)
MV1bo
Figure 5-2 Motion vector estimator 1

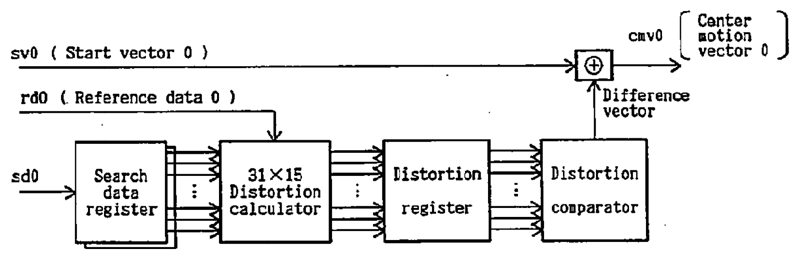


Figure 5-3 MVD1 (Motion vector detector 1)

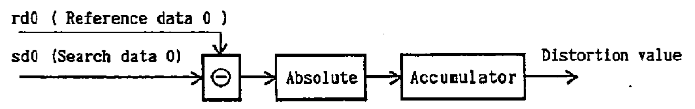


Figure 5-4 One of 31×15 Distortion calculators

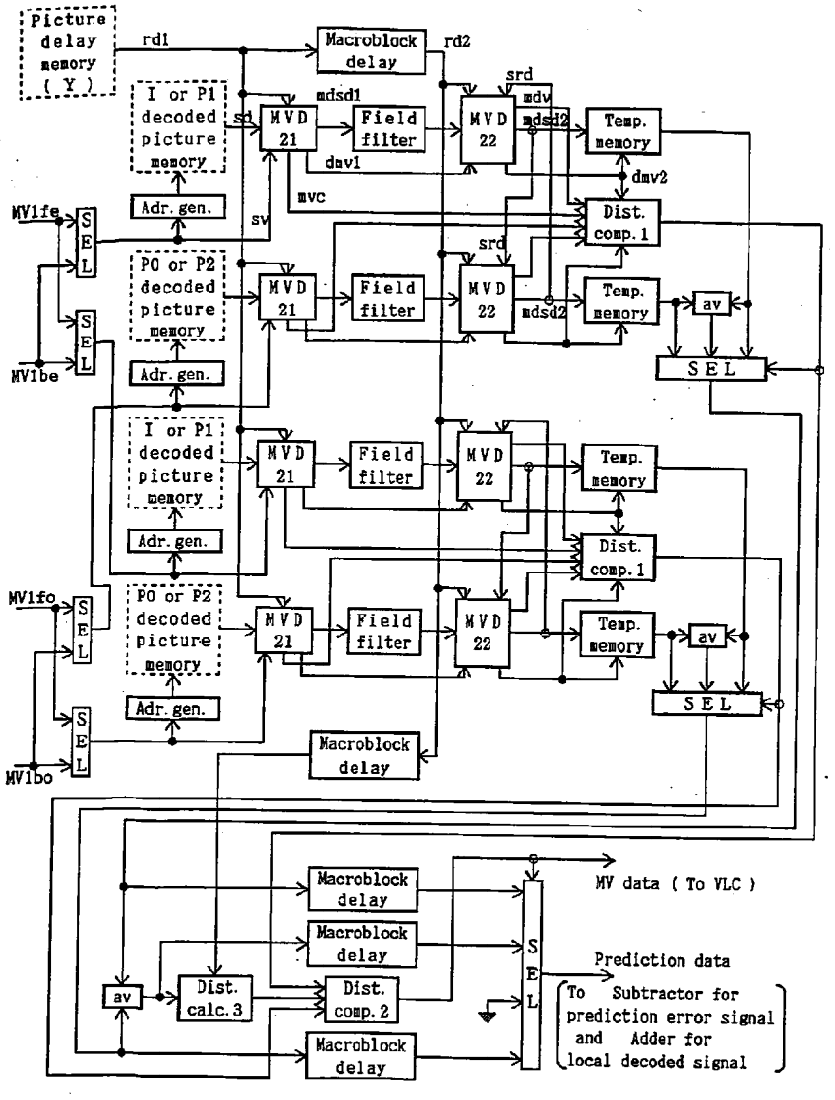


Figure 5-5 Motion vector estimator 2 with adaptive predictor

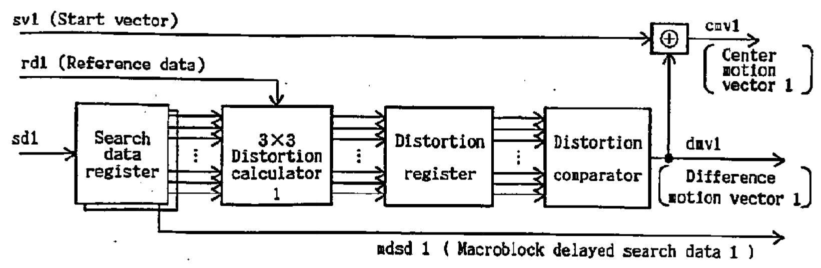


Figure 5-6 MVD21 (Motion vector detector 21)

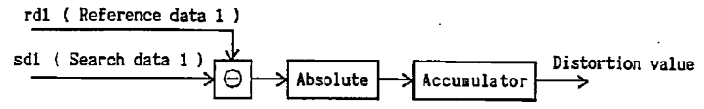


Figure 5-7 One of 3×3 Distortion calculators 1

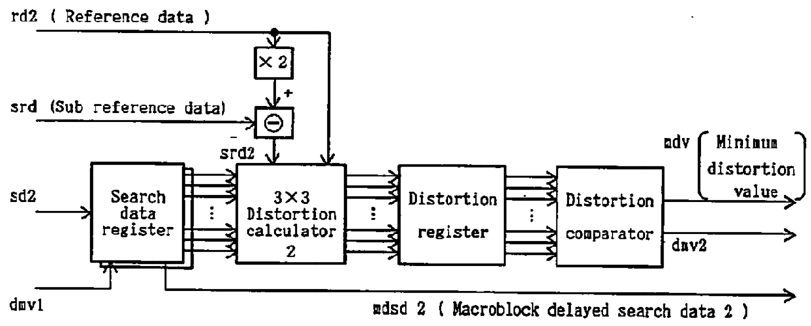


Figure 5-8 MVD22 (Motion vector detector 22)

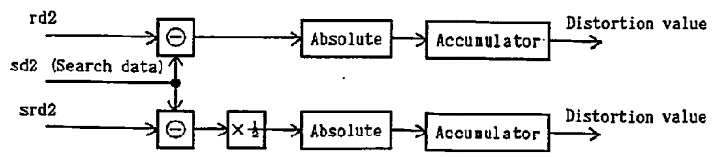


Figure 5-9 One of 3×3 Distortion calculators 2

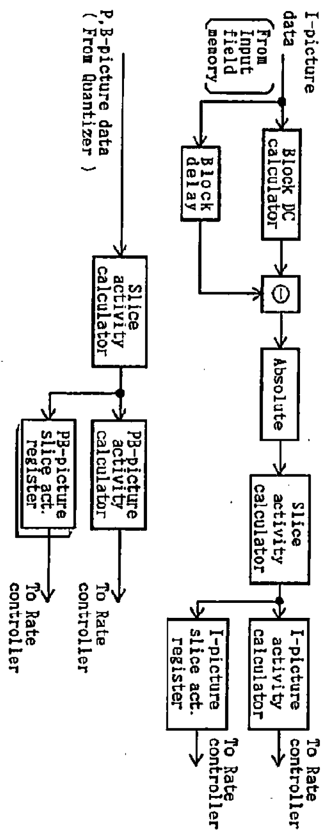

I-picture data
From input field memory
Block DC calculator
Block delay
Absolute
Slice activity calculator
I-picture activity calculator
I-picture slice act. register
To Rate controller
To Rate controller
P,B-picture data (From Quantizer)
Slice activity calculator
PB-picture activity calculator
PB-picture slice act. register
To Rate controller
To Rate controller

Figure 5-10 Activity calculator

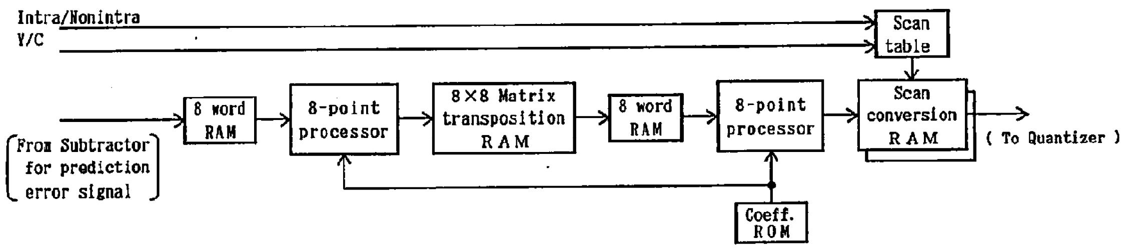


Figure 5-11 DCT

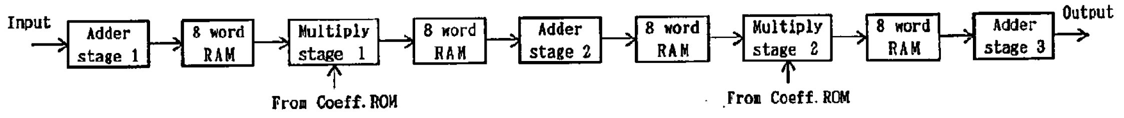


Figure 5-12 8-point processor

MS-MOTO_752_0001230359

MS-MOTO_1823_00000720879

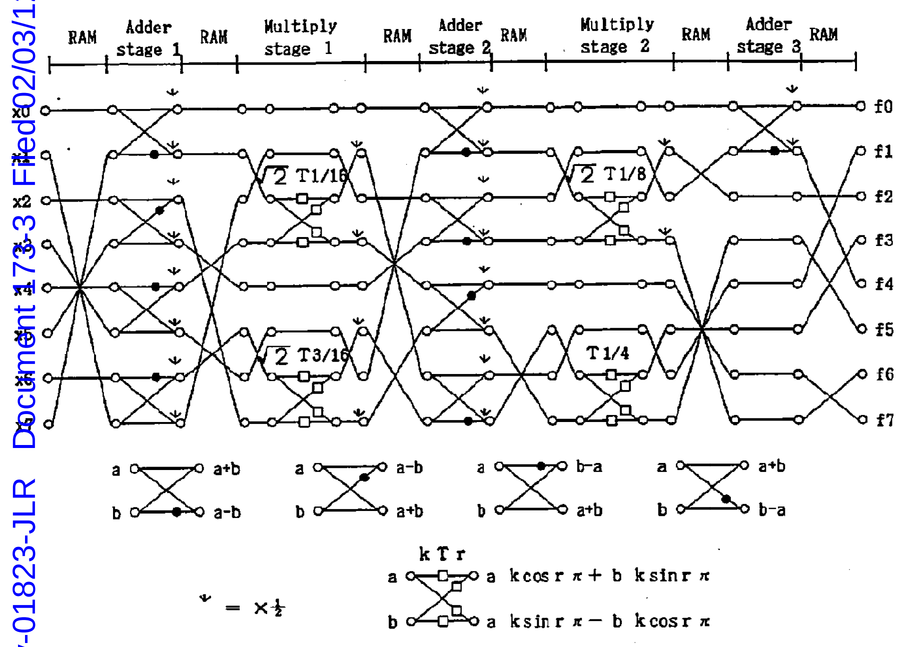


Figure 5-13 8-point DCT signal flowgraph

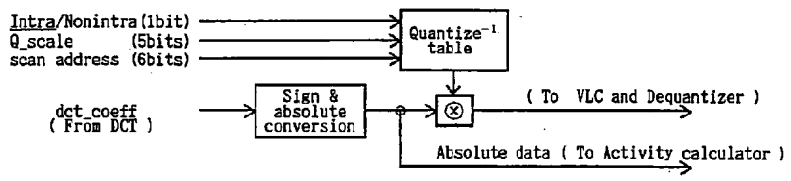


Figure 5-14 Quantizer

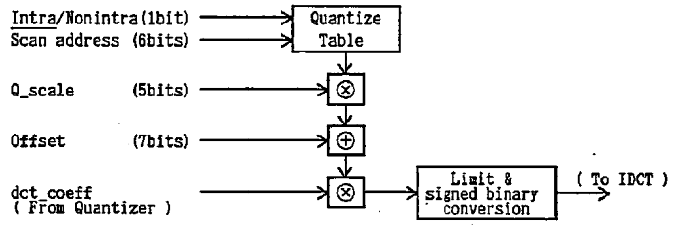


Figure 5-15 Dequantizer